Treball de Fi de Grau

Enginyeria en Tecnologies Industrials

Disseny i construcció d'un robot control·lat per veu

MEMÒRIA

Autor: Adrián Fontal Chacón

Director: Cecilio Angulo Bahón

Data: Febrer 2020

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resum

L'objectiu d'aquest treball és el disseny i la construcció d'un prototip de braç robot amb dos graus de llibertat control·lat per veu. Per això, s'estudiarà els materials necessaris per assolir aquest objectiu, les accions que haurà de fer el robot i les ordres per les que les haurà de efectuar, a més de possibles millores d'aquest i estudis del que pot aportar el meu projecte.

Aquest projecte es farà de zero, és a dir, tota la programació, estudi de moviments i construcció estarà dissenyada per mi. Llavors, s'haurà d'afrontar els reptes que suposa fer un projecte des de zero, com per exemple buscar tota la informació necessària i aprendre els diferents tipus de coneixements que necessitaré per efectuar el projecte.

Així, el treball es composarà d'una part teòrica on s'estudiarà des de la part més bàsica el que és un robot, les parts que el poden formar i un estudi més específic d'aquestes parts, i després l'estudi del que serà el disseny, construcció i programació final del robot amb les opcions escollides.

Sumari

Índex de figures

Índex de taules

Capítol 1

Introducció

La robòtica [22] és la branca de l'enginyeria que tracta el disseny, construcció, operació, estructura, manufactura i aplicació de robots. Per tal fet, és necessari combinar diverses competències de l'enginyeria, tals com mecànica, electrònica o informàtica, entre d'altres. Neix de la necessitat de L'ésser humà de materialitzar el desig de crear quelcom a la seva semblança i que, al mateix temps, el descarreguessin dels treballs més durs.

1.1 Classificació de robots

Segons diferents criteris de estudi, es classifiquen els robots de diverses formes:

1.1.1 Segons la generació

1ª Generació

Robots manipuladors. Sistemes mecànics multifuncionals amb sistemes senzills de control.

2ª Generació

Robots d'aprenentatge. Copien un seguit de moviments prèviament realitzats per un operador humà.

3ª Generació

Robots amb control sensoritzat. Un controlador executa les ordres d'un programa i les envia a la màquina per a que les realitzi.

1.1.2 Segons l'estructura

Multiarticulat

Robots que tenen la principal característica de ser "sedentaris" (tot i que poden tenir un sistema de guia per ampliar el moviment), pel qual els seus moviments estan destinats a localitzar-se en una zona de treball delimitada.

Mòbils

Robots amb capacitat de desplaçament basada en carros o plataformes, dotats de sistemes locomotor per rodes. Es solen utilitzar per assegurar peces d'un punt a un altre en cadenes de fabricació.

Androides

Robots que intenten reproduir la forma o el comportament humà. Actualment els androides existents són poc evolucionats i sense una utilitat pràctica, destinats principalment a l'estudi i l'experimentació.

Zoomòrfics

Robots que imiten en el seu comportament a éssers vius. S'estudia la utilització d'aquests tipus de robots als camps d'estudi de l'exploració espacial o a l'estudi dels volcans.

Híbrids

Robots que no es poden classificar a cap dels apartats anteriors essent combinacions d'aquests.

1.2 Requeriments previs

Per poder realitzar aquest treball, es necessitarà tenir una base de programació per poder manipular el projecte.

També es requerirà de certs coneixements d'electrònica i electricitat, per tal de ser capaç de dissenyar i muntar un circuit electrònic i elèctric que pugui respondre a la programació que es decideixi.

Per últim, serà necessari conèixer certs aspectes d'anàlisi de mecanismes per tal de poder dissenyar o, en el cas d'utilitzar un disseny ja existent, control·lar una estructura mòbil.

1.3 Metodologia

Pel que fa la metodologia, s'ha seguit la planificació detallada al diagrama de Gantt de la Figura 1.1.

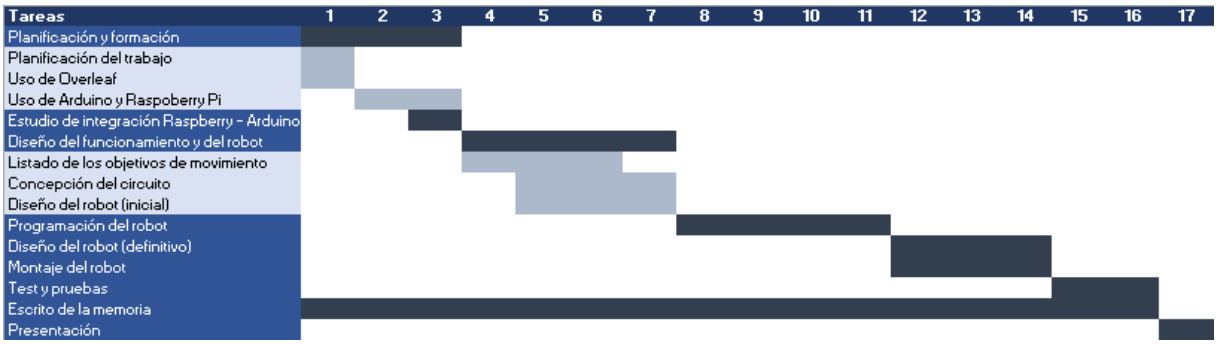

Figura 1.1: Diagrama de Gantt de les 17 setmanes de duració del projecte.

Primerament, es va estudiar com utilitzar les plataformes d'*Arduino* i *Overleaf*.

Seguidament, es va investigar com aplicar les opcions que tenia Arduino a la idea del projecte.

Després d'això, va venir la part més complicada: dissenyar el que es faria amb el robot, com ho faria i, per tant, com hauria de ser per realitzar aquestes accions.

Per tal de programar el prototip, primerament es va haver de dissenyar un braç robot simple que fos capaç de moure's de la manera que es pretenia. Després d'això, es va decidir com es fabricarien les peces del robot i com s'unirien entre elles.

Una vegada construït el robot, es va procedir a programar-lo.

Finalment, la tasca va ser provar el programa decidit i anar afinant-lo per tal que assolis els objectius el millor possible.

Durant tot aquest projecte, a més, s'ha anat redactant aquesta memòria.

1.4 Objectius

L'objectiu d'aquest projecte és construir un prototip de braç robot a ser utilitzat de forma col·laborativa (*cobot* [23])

Tot i així, l'objectiu no és crear un *cobot* complexe com els que ja existeixen a les fàbriques. La idea és dissenyar un robot al a un nivell bàsic i comprovar si el control per veu és quelcom útil, essent conscients de les limitacions que es té a l'hora dels coneixements necessaris.

Per tant, l'objectiu final es estudiar com millorar el projecte per tal de poder-ho fer una realitat a la indústria.

1.5 Abast

En el projecte es vol analitzar com de bona és la resposta del robot col·laboratiu construït donant-li ordres bàsiques i veient si es comporta correcta i ràpidament.

Dintre dels objectius d'abast d'aquest projecte, no es planteja millorar els *cobots* existents, si no veure si aquesta idea podria ser útil.

Capítol 2

Conceptes bàsics

En aquest capítol, abans d'entrar en matèria amb el treball, s'exposarà de forma general les opcions existents per poder control·lar el robot del que consta aquest projecte, a més dels diferents modes que es tenen per atorgar-li mobilitat. Així, es podrà tenir una idea de tot el ventall d'opcions que tenim i poder justificar l'elecció final.

2.1 Control·ladors

Per tal de poder control·lar el comportament del robot dissenyat, serà necessari disposar d'un control·lador o placa de control. D'aquests hi ha de molt diferents, amb més o menys dificultat, però en aquest projecte s'estudiarà tres d'aquests, ja que o són els més utilitzats o es té confiança al haver-ho utilitzat prèviament.

2.1.1 Arduino

Arduino [16] és una companyia de desenvolupament *hardware* i *software* que dissenya plaques de desenvolupament de hardware, com la que s'observa a la Figura 2.1, per tal de construir dispositius digitals i interactius que poden detectar i control·lar objectes del món real.

L'objectiu és poder apropar i facilitar l'ús de l'electrònica i programació de sistemes encastats en projectes multidisciplinars.

Aquestes plaques utilitzen un seguit de microcontrol·ladors i microprocessadors, connectats sota la configuració de "sístema mínim" (circuit bàsic d'operació del microcontrol·lador) sobre una placa de circuit imprès al que se li poden connectar plaques d'expansió (*shields*) mitjançant els ports d'entrada o sortida disponibles. Això fa que la placa guanyi molta versatilitat, ja que es poden implementar mòduls tals com sensors de tacte, ultrasons, així com mòduls per control·lar motors de corrent contínua, entre d'altres.

Arduino és un *hardware* lliure. Això vol dir que tothom pot experimentar amb Arduino i distribuir els seus dissenys. Aquests es distribueixen sota llicència *Creative Commons Attribution Share-Alike 2.5*, i tots estan disponibles a la pàgina web d'Arduino.

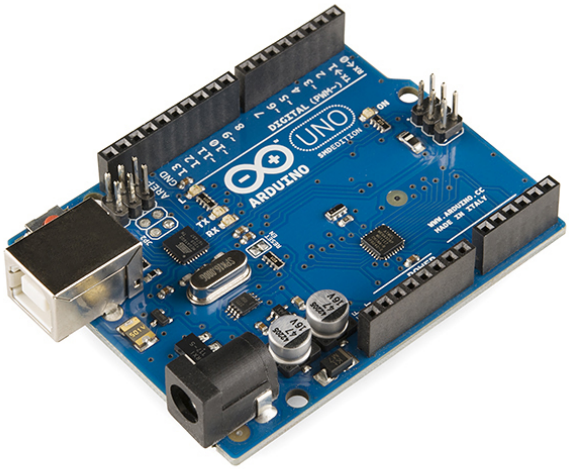

Figura 2.1: Placa Arduino UNO

2.1.2 Raspberry Pi

Raspberry Pi [21] (veure Figura 2.2) és un **ordinador de placa reduïda**, és a dir, un ordinador complet en un sol circuit, que té tot el necessari per funcionar a la placa base al contrari que els ordinadors convencionals, que tenen una placa base i la resta de funcions les fan altres components.

Tot i no expressar que és un *hardware* lliure, a la web exposen que qualsevol pot ser distribuïdor de tarjetes *Raspberry Pi*, pel qual s'entén que és un producte amb marca registrada però de lliure ús. D'altra manera, el *software* sí és lliure, utilitzant un sistema operatiu anomenat *Raspbian*, una adaptació del sistema operatiu *Debian* (basat en els sistemes operatius *GNU/Linux*), tot i que pot utilitzar altres sistemes operatius tals com Windows 10.

En qualsevol de les versions que es troben a la venda, s'inclou un processador *Broadcom*, memòria *RAM*, *GPU*, ports *USB*, *HDMI*, *Ethernet*, 40 pins *GPIO* i un connector per a càmera.

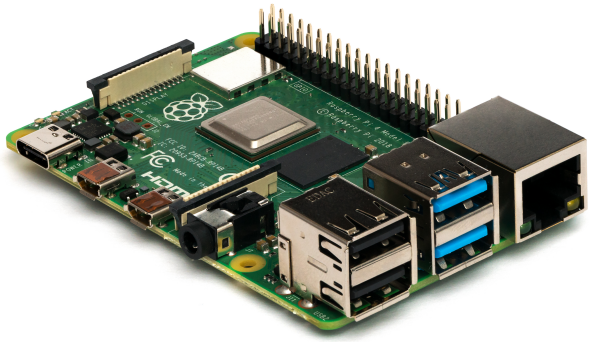

Figura 2.2: Placa Raspberry Pi, microprocessador.

2.1.3 PICkit

PICkit [9] (veure Figura 2.3) és una família de programadors per microcontrol·ladors PIC.

És una eina de baix cost amb una interfície fàcil d'utilitzar. El circuit de depuració de l'eina fa que el programa s'executi, pari i avanci pas per pas mentre el microcontrol·lador PIC està integrat a l'aplicació.

Figura 2.3: PICkit, programador de microcontrol·ladors Microchip PIC.

2.2 Programació

Els sistemes que s'han mostrat a l'apartat anterior es basen en microcontrol·ladors. Per tal de poder utilitzar-los per control·lar un sistema, s'haurà d'utilitzar un llenguatge de programació adient pel control·lador corresponent.

2.2.1 Arduino

El codi de programació de les plaques Arduino és un codi basat en el codi C++, pel qual té gran part de les seves característiques.

C++ [2] [17] és un llenguatge de programació dissenyat a mitjans dels 80 per Bjarne Stroustrup, concebut per estendre la programació en C amb mecanismes que permetessin la manipulació d'objectes. Es diu que és un llenguatge *multiparadigma* (els programadors no s'han d'adaptar a un estil particular de programació, ja que permet diversos estils de programació).

En el cas d'Arduino, el programa es basa en dos blocs: un bloc `setup` i un `loop`. El bloc `setup` s'encarrega de definir i inicialitzar les variables que s'utilitzaran en el programa; al ser un codi C++, les variables s'han de definir indicant del tipus que són. El bloc `loop` s'encarrega del programa com a tal, és a dir, en aquesta part s'escriurà el que ha de fer el sistema a control·lar. Abans d'aquests dos blocs, però, s'haurà d'inicialitzar les diverses llibreries que es necessitaran pel programa.

A la Figura 2.4 es pot veure el procés que segueix el codi, des de que s'escriu fins que arriba al microcontrol·lador.

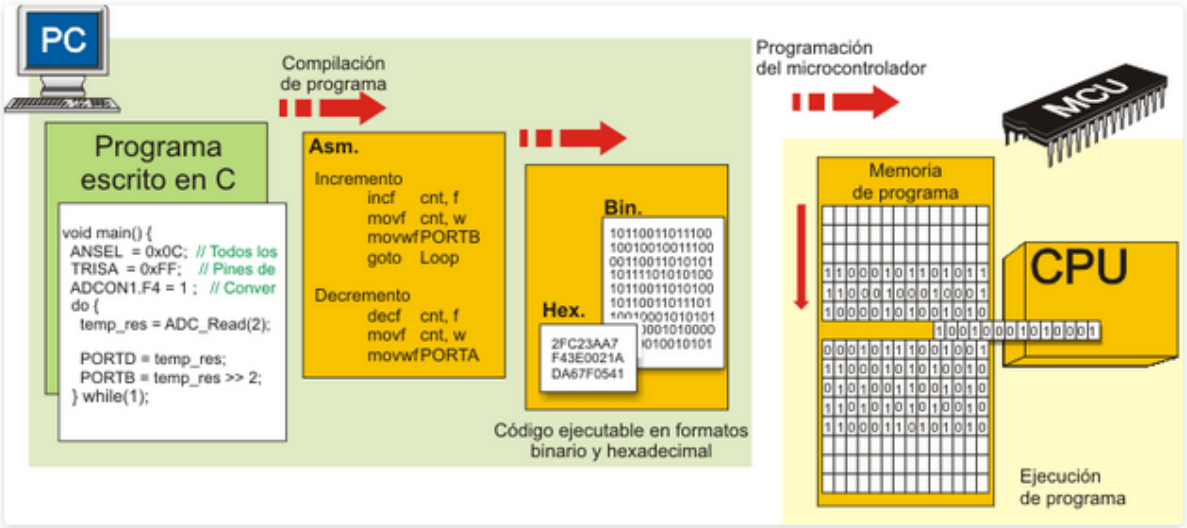

Figura 2.4: Programar en C

2.2.2 Python

En el cas de la Raspberry Pi, al funcionar en un entorn *Linux*, el principal mode de programació que utilitza (ja que és el mode natiu) es Python.

Python [20] és un llenguatge de programació creat al final de la dècada dels vuitanta per Guido van Rossum, com a successor del llenguatge de programació ABC, i administrat per la Python Software Foundation.

El llenguatge, tal com C++, és del tipus *multiparadigma*,. És un llenguatge interpretat, dinàmic i multiplataforma.

Una de les principals característiques d'aquest llenguatge és que s'ha dissenyat per ser llegit amb facilitat. Per exemple, els operadors lògics `!`, `||` i `&&`, característics del codi C, són `NOT`, `OR` i `AND` respectivament. A més, el contingut dels blocs de codi es delimiten utilitzant espais o tabuladors, en lloc de utilitzar un conjunt de caràcters, normalment entre claus {}. A la Figura 2.5 es poden observar algunes diferències remarcables entre els codis C i Python.

```c
int factorial(int x)
{
    if (x < 0 || x % 1 != 0) {
        printf("x debe ser un numero entero mayor o igual a 0");
        return -1; //Error
    }
    if (x == 0) {
        return 1;
    }
    return x * factorial(x - 1);
}
```

```python
def factorial(x):
    assert x >= 0 and x % 1 == 0, "x debe ser un entero mayor o igual a 0."
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)
```

Figura 2.5: Exemple estructura C vs Estructura Python.

Entre d'altres diferències remarcables, la declaració de variables es fa de forma dinàmica, és a dir, no s'ha d'especificar el tipus de dada de la variable.

2.2.3 C

A més dels anteriors llenguatges, s'estudiarà C [18], ja que és el llenguatge utilitzat pel control·lador PicKit, tot i que tindrà moltes de les característiques de C++.

C és un llenguatge desenvolupat per Dennis Ritchie entre els anys 69 i 72, com a una evolució de l'anterior llenguatge B.

L'objectiu que es van marcar al desenvolupar aquest llenguatge va ser que fossin necessàries poques instruccions per traduir cada element del llenguatge, amb poc temps d'execució. A més, es tracta d'un estil de programació de baix nivell, pel qual es poden desenvolupar compiladors d'aquest llenguatge fàcilment.

2.3 Selecció

En aquest punt, s'hauria de decidir la plataforma que s'utilitzarà per control·lar el robot, a més del tipus de robot que serà. Després d'això, es podrà començar a pensar que s'utilitzarà per programar i construir el projecte.

2.3.1 Control·lador

Aquest serà l'element que permetrà control·lar el robot. Amb el nivell de coneixements de programació que es tenen al principi del projecte, s'haurà d'escollir un control·lador que sigui de fàcil aprenentatge, a més d'un que tingui molta documentació (tant física com *on-line*) a l'abast, i el qual la programació i el muntatge sigui simple. També serà important que sigui fàcil d'aconseguir.

Per tal, la decisió del controlador ha sigut **Arduino**, ja que al laboratori hi ha molta disposició de llibres i informació per programar en Arduino, molts mòduls extra per utilitzar compatibles amb Arduino, a més de la pròpia placa.

2.3.2 Tipus de robot

Una vegada escollit el mètode de programació, s'havia d'escollir el tipus i forma del robot, per tal de poder decidir el seu disseny.

En aquest cas, el robot serà un braç robot tipus *cobot*. Aquest tipus de robot és un robot de 3ª generació multiarticulat.

Les diferències entre un braç robot d'indústria i un *cobot* són significants. Els braços industrials, per seguretat, han d'estar envoltats de tanques, ja que tant pel seu pes com per la quantitat de força amb la que operen no poden estar en contacte directe amb l'artefacte. En canvi, el robot col·laboratiu està pensat per treballar amb l'operari, i està pensat amb un seguit de sensors que asseguren la seguretat de qualsevol que treballi amb ell.

A més, el braç industrial està fixe en un lloc. En canvi, el robot col·laboratiu, al ser de dimensions més reduïdes, es pot anar movent per la fàbrica fàcilment segons les necessitats.

Llavors, el prototip que es dissenyarà serà un robot col·laboratiu de 2 eixos, 3 articulacions i una pinça senzilla.

Capítol 3

Elements per la construcció del robot

Aquest capítol es centrarà en explicar totes les parts que es necessitaran per dur a terme el projecte, tal com parts mecàniques o motors, entre d'altres.

Per tal de fer un bon estudi, primerament es començarà per les parts mecàniques del braç robot, i es finalitzarà amb el necessari per a la programació.

3.1 Elements mecànics

Dintre dels elements mecànics que composaran el robot, tenim dos ben definits: servomotors i elements mòbils del braç.

3.1.1 Servomotors

Per poder construir el braç que s'ha dissenyat, es necessitarà un total de 4 servomotors, 2 del tipus **MG995** (Figura 3.1), 1 del tipus **SM-S2309S** (Figura 3.2) i 1 del tipus **VMA600** (Figura 3.3). Els del primer tipus seran per les articulacions del braç, i l'altra s'utilitzarà pel moviment de la pinça.

Servomotor MG995

El servomotor MG995 [24], que es pot observar a la Figura 3.1, és un servomotor amb engranatges metàl·lics i un moviment de 180º compatible amb *Arduino*. Els engranatges fan que tingui una gran potència, pel qual és idoni per poder aixecar qualsevol dels pesos que es tindran que moure en aquest projecte.

Les seves especificacions tècniques són:

- **Mida:** 40×19×43 mm
- **Pes:** 69 grams

- **Velocitat de rotació (4.8V):** 0,17 sec / 60º

- **Força (4.8V):** 1,2748645 N/m

Figura 3.1: Servomotor MG995.

Servomotor SM-S2309S

De forma similar a l'altre model,el servomotor SM-S2309S [13] té la seva pròpia circuiteria per tal de funcionar amb un microcontrol·lador de forma individual, tal com es pot veure a la Figura 3.2. Com a diferència, és molt més petit, pel què serà una bona incorporació per la pinça o l'articulació del canell.

Les seves especificacions tècniques són:

- **Mida:** 22.2×11.6×21.5 mm

- **Pes:** 9 grams

- **Velocitat de rotació (4.8V):** 0,12 sec / 60º

- **Força:** 0.0980665 N/m

Figura 3.2: SM-S2309S

Servomotor VMA600

El servomotor VMA600 [15] és un servomotor petit té unes característiques similars, pel qual també serà útil per la pinça o pel canell.

Les seves especificacions tècniques són:

- **Mida:** 22.7×11.6×21.5 mm
- **Pes:** 9 grams
- **Velocitat de rotació (6V):** 0,09 sec / 60º
- **Força:** 0.1372931 N/m

Figura 3.3: Servomotor VMA600.

3.2 Elements electrònics

Els elements electrònics que s'utilitzaran per a aquest projecte són una placa *Arduino UNO*, un mòdul de control per veu *ELECHOUSE V3*, una *protoboard* i un alimentador de 9V 2A.

3.2.1 Arduino UNO

La versió UNO és, segons la pròpia pàgina web d'Arduino, la millor placa per començar amb l'electrònica i la programació, ja que és la placa més utilitzada i, per tant, la millor documentada.

Arduino UNO [3] (Figura 2.1) és un microcontrol·lador basat en el *ATmega328P*. Consta de 14 pins digitals (6 dels quals es poden utilitzar com a sortides PWM), 6 pins analògics, un cristall de quars de 16 MHz, connexió USB, un *jack* de potència, una entrada ICSP i un botó *reset*.

Les especificacions tècniques de la placa són:

- **Microcontrol·lador:** *ATmega328P*
- **Voltatge de funcionament:** 5V
- **Voltatge (recomenat):** 7-12V
- **Voltatge (límit):** 6-20V
- **Pins digitals I/O:** 14 (6 amb sortida PWM)
- **Corrent contínua als pins I/O:** 20 mA
- **Corrent contínua pels pins de 3.3V:** 50 mA
- **Memòria flash:** 32 KB (*ATmega328P*) dels quals 0.5 KB s'usen per arrancar la placa.
- **SRAM:** 2 KB (*ATmega328P*)
- **EEPROM:** 1 KB (*ATmega328P*)
- **Clock Speed:** 16 MHz
- **LED BUILTIN:** 13
- **Mida:** 68.6×53.4 mm

- **Pes:** 25 g

3.2.2 ELECHOUSE V3

El mòdul de reconeixement de veu de ELECHOUSE [7] (Figura 3.4) és una petita placa que ofereix un reconeixement de veu de forma fàcil, ja que porta la seva pròpia llibreria que facilita el seu us.

Aquesta placa reconeix un màxim de 80 comandaments de veu, però només poden funcionar 7 a la vegada. Per funcionar, l'usuari haurà d'entrenar el mòdul abans de poder utilitzar-se, però l'usuari pot utilitzar qualsevol so com a comandament. Per tant, és necessari un micròfon per utilitzar-se.

Les especificacions tècniques són:

- **Voltatge:** 4.5-5.5V
- **Corrent:** <40mA
- **Interfície digital:** 5V TTL level per a interfaç UART i GPIO
- **Interfície analògica:** Connector de micròfon de 3.5mm + pin per micròfon
- **Mida:** 31mm×50mm
- **Precisió del reconeixement:** 99% en un entorn ideal.

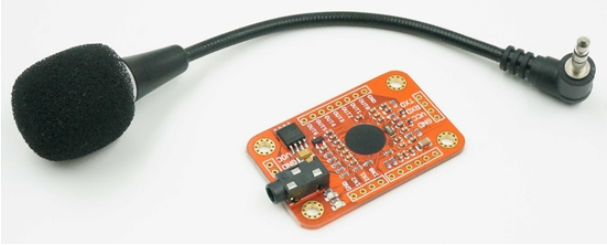

Figura 3.4: ELECHOUSE V3

3.2.3 Protoboard

Una placa de proves o *protoboard* [19] (Figura 3.5 (dreta)) és una taula amb un seguit de forats que estan connectats elèctricament entre si internament, utilitzant el patró de línies que es pot veure també a la Figura 3.5 (esquerra). D'aquesta manera, es pot connectar elements electrònics de forma directa o utilitzant cables.

Les parts d'aquesta taula són molt simples. Una part és un material aïllant, usualment plàstic, i un material conductor que connecta els patrons de línies.

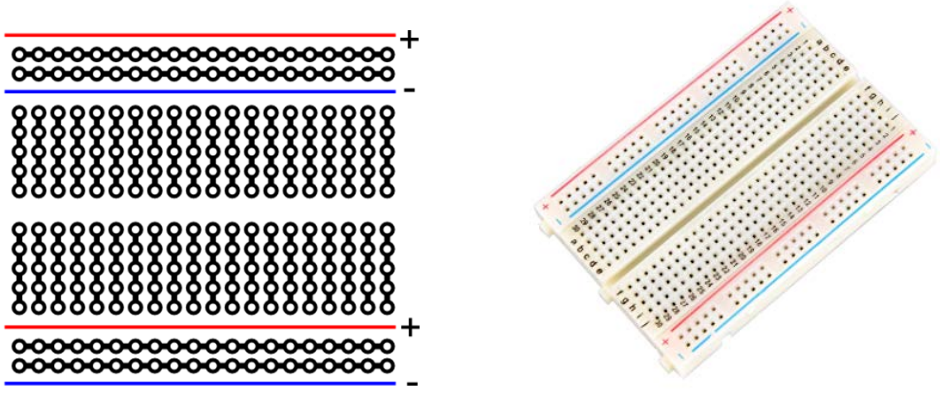

Figura 3.5: Protoboard

3.2.4 Alimentador commutat 9V 2A Euroconnex

Per aportar energia al conjunt, es necessitarà un alimentador extern a l'*Arduino*. En aquest cas, s'utilitzarà un alimentador de la marca Euroconnex [8] (Figura 3.6) que proveirà 9V al conjunt (dintre dels paràmetres de funcionament d'*Arduino UNO*) i 2A (suficients per poder alimentar tot el que ha de moure el projecte).

El alimentador va connectat a la corrent, i mitjançant un transformador passa de corrent alterna (tensió d'entrada 100/240V a 50/60Hz) a corrent continu.

La forma de connectar l'alimentador a l'*Arduino* és amb una clavilla de 5,5×2,1 mm a l'entrada de la placa.

Figura 3.6: Alimentador

Capítol 4

Construcció del braç robot

En aquest capítol s'exposarà el procés de disseny i construcció del braç robot, a més dels moviments que farà. Per tant, es veurà els plànols, els materials utilitzats i com s'han fabricat les peces.

4.1 Disseny del braç

El robot consisteix en 3 peces separades que formen el braç, una peça més complexa que forma la pinça i una base on suportar-lo i tenir la circuiteria necessària.

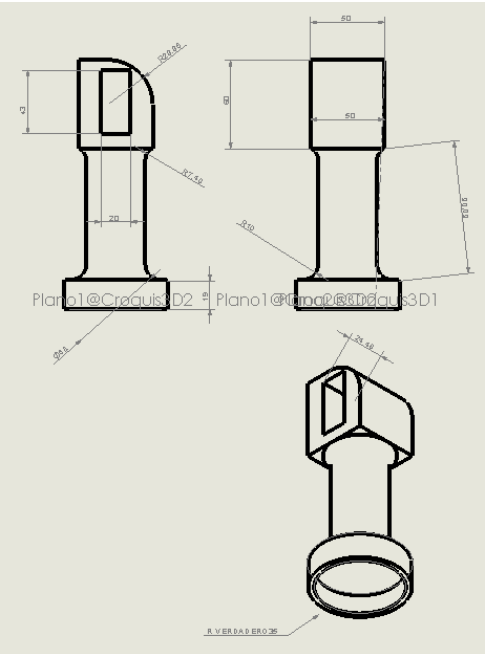

Figura 4.1: Base del braç.

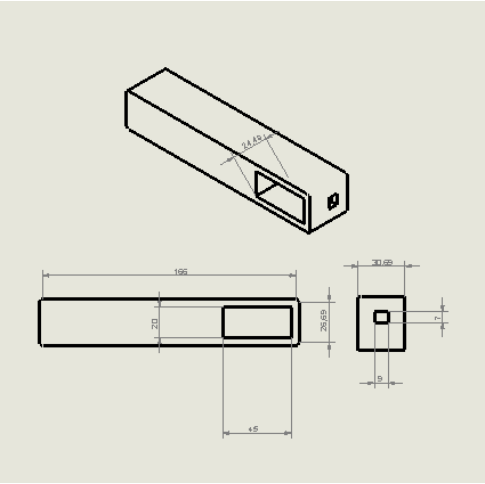

Figura 4.2: Braç espatlla - colze

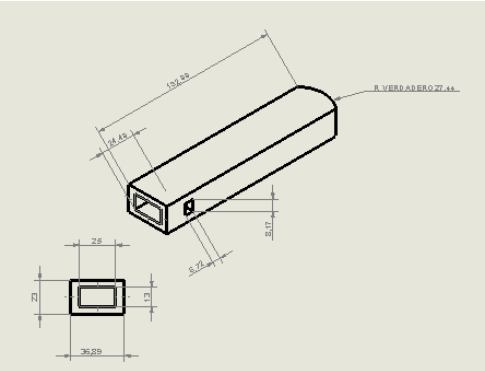

Figura 4.3: Braç colze - canell

Com es pot veure a les imatges (Figura 4.1, Figura 4.2 i Figura 4.3), els elements del braç són elements molt simples. Les dues parts mòbils són bastant similars, i la base suportarà la resta del braç, que per aquest mateix motiu està més reforçada.

Totes les parts del braç tenen una cavitat expressament feta per encabir el servomotor que mourà la següent part, de tal manera que no és necessari cap element d'unió entre el servomotor i el braç ja que queda perfectament encaixat en aquest.

A més, es pot observar unes petites incisions comuns a totes les parts dels braç, les quals són les sortides del cablejat del servomotor.

4.1.1 Pinça

En aquest cas, primerament es va pensar en un disseny de pinça utilitzant un servomotor petit. Per realitzar aquesta idea, es va dissenyar una pinça personalitzada. Tot i així, la dificultat de posar en comú tantes peces petites va fer que la decisió fos utilitzar un

disseny d'Internet [12] pensat per un servomotor petit, de la mida del que ja es tenia pel primer disseny de pinça. A la Figura 4.4 es presenten els plànols de la pinça.

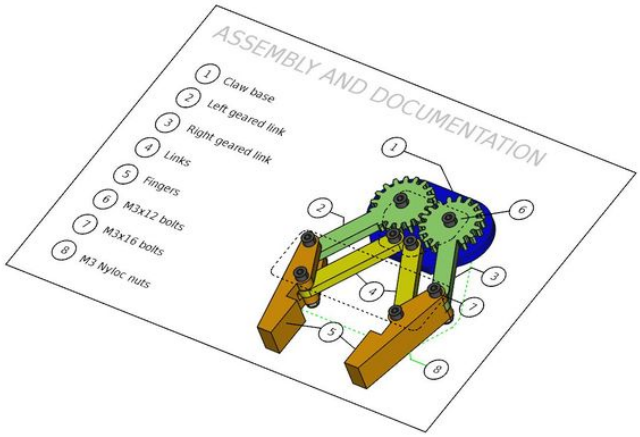

Figura 4.4: Pinça

4.2 Fabricació de les peces

La fabricació de les peces s'ha fet amb una impressora 3D, marca Creality, model Ender 3 Pro [4].

El programa utilitzat per fer funcionar la impressora és *Ultimaker Cura V4.2.1* (veure Figura 4.5). Aquest programa permet la visualització de les peces 3D i la modificació de les seves característiques a l'hora de fer la impressió. El plàstic que s'ha utilitzat per fer la impressió de les peces ha sigut PLA (àcid polilàctic), un polímer molt utilitzat en la impressió 3D.

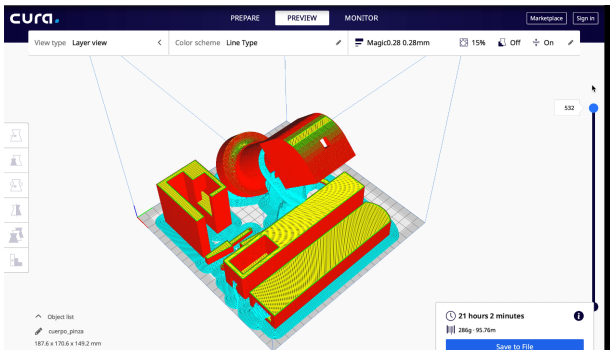

Figura 4.5: Pantalla d'impressió a la impressora 3D.

Entre les característiques que s'han de tenir en compte a l'hora de fer la impressió tenim la qualitat, l'ompliment, la velocitat, la temperatura i el filament.

Per control·lar la qualitat, tindrem en compte les següents característiques:

- L'altura de capa [5], la qual és la separació entre capa a capa de la impressió. Es mesura en micres, i la separació marcarà de quina forma es veu la peça: quanta menys separació entre capes, menys visible serà que està fet per capes. En aquest cas, l'altura de capa ha estat 28000 micres.
- La manera en que esta feta la peça es *brim* [10]. Això crea una capa al voltant de la peça al principi, així evitant errors a l'inici de la impressió.

També s'hauran de tenir en compte la velocitat de impressió i la temperatura.

- La velocitat d'impressió fa que quant més lenta funcioni la impressora, més qualitat tindrà la peça. En aquest cas, la velocitat d'impressió ha sigut de 70 mm/s.
- La temperatura d'impressió haurà de ser compatible amb el plàstic utilitzat. També s'haurà de tenir en compte la temperatura del llit, ja que farà que el plàstic es quedi enganxat pel calor. Llavors la peça queda immòbil i es redueixen les vibracions al fer la impressió.

Finalment, també s'haurà de tenir en compte l'ompliment de la peça, amb els paràmetres del grossor de la capa, el tipus de farcit i la densitat de farcit.

- El gruix de les capes es mesurarà en mil·límetres.
- La densitat del farcit serà la quantitat de plàstic que trobem entre les capes de la figura. En el cas d'aquestes peces, hi haurà un farcit tipus grid del 15%.

A més, hi haurà peces que necessitin un suport addicional al ser impreses. Això significa que en les zones de més risc de deformació de la peça s'aplica un plàstic de menys qualitat per tal d'evitar aquesta deformació.

Finalment, alguns criteris de impressió més [11]:

- **Diàmetre de la boqueta d'impressió:** 0.4mm
- **Temps d'impressió total:** 21h 2m
- **Pes total:** 286g
- **Metres de material total:** 95.76m

Capítol 5

Programació i muntatge

En aquest apartat es farà un estudi de quins moviments farà el braç, com es programarà el robot per tal de realitzar aquests moviments i el muntatge tant del circuït com del braç.

5.1 Moviments

En aquesta part de la memòria s'especificaran els moviments que haurà de fer el nostre projecte, juntament amb les comandes de veu a les quals haurà de respondre.

Comandament	Moviment
Puja	El braç va cap a la seva configuració superior.
Obre	Obre la pinça.
A sota	El braç va cap a la seva configuració inferior.
Tanca	Tanca la pinça.
Apaga	El braç es recull i es desactiva.
Rosca	Agafa l'objecte, gira, el solta, gira a l'altre sentit, i repeteix el proces quatre vegades.
Desenrosca	El mateix que el comandament rosca, però cap a l'altre sentit.

Taula 5.1: Comandament i moviment

Com es pot veure a la Taula 5.1, són moviments molt senzills que incorporen des del moviment d'una sola articulació fins al moviment de totes. En el cas que es mogui més d'una articulació, s'ha decidit que aquestes es moguin una a la vegada, amb un temps entre un moviment i un altre.

5.2 Programació

Com ja s'ha comentat, per programar el projecte s'ha utilitzat el llenguatge C++, utilitzant la interfície de programació que ofereix Arduino.

Per fer la programació del braç, s'ha reutilitzat un dels codis exemple que dóna la llibreria del mòdul de reconeixement de veu. Aquest està pensat per al control d'un dispositiu LED, però coneixent-lo i amb unes petites modificacions es pot aconseguir que faci el que es necessita pel projecte.

El codi utilitzat pel projecte està recollit a l'Apèndix A.

Per entendre com funciona la base del codi, la programació dels servomotors, primer s'haurà d'entendre com funciona un servomotor.

Els servomotors que s'utilitzen en aquest projecte són servomotors de rang de gir limitat [14], es a dir, només poden anar de 0º a 180º.

Aquests servomotors funciona amb una senyal `PWM`, en aquest cas proveïda pels pins digitals de la placa Arduino, que rep el circuit electrònic del servo. Com es pot veure a la Figura 5.1, segons el temps en que la senyal està a nivell alt dintre del temps de senyal, el servomotor es mourà a un angle o un altre. Per tant, quan al programa s'especifica l'angle al qual es vol anar, el que realment s'està especificant és el temps que la senyal estarà a nivell alt.

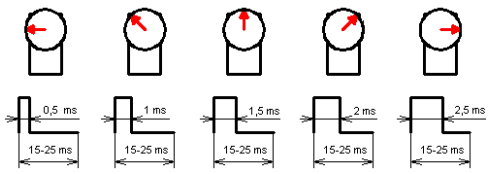

Figura 5.1: Instruccions de muntatge de la pinça

El codi ha anat sofrint variacions en la seva complexitat durant la programació, tant per idees que han anat sorgint com a causa del propi aprenentatge.

Primerament, la idea inicial va ser que els servomotors utilitzessin la funció `write` directament. Això feia que els moviments del braç fossin bruscos i fins i tot donava problemes als servos. Per solucionar aquest problema, s'ha inicialitzat una sentència `while` que fa moure el servomotor grau a grau amb un cert retard entre aquests, suficientment gran perquè el moviment no sigui brusc però suficientment petit perquè el moviment es vegi fluid.

Un altre problema a solucionar era conèixer la posició a cada moment per saber quants graus s'havien de moure els servomotors per realitzar les accions pertinents. Primerament, es va pensar en que abans de cada moviment es tornés a una posició per defecte, anomenada "casa", per saber la posició exacta. Això feia que, a més d'ocupar innecessàriament un comandament, feia que la utilitat del braç es reduís, ja que la continuïtat de moviment es veuria afectada. Per arreglar-ho, es van introduir un seguit de variables per controlar la posició de cada servomotor al final de cada moviment. A més, es van afegir un seguit de clàusules `if` per tal de, controlant si la posició del servomotor era més gran o més petita que la posició final, l'angle augmentaria o disminuiria.

Apart d'això, el codi s'ha mantingut simple, utilitzant clàusules `case` per diferenciar què haurà de fer el robot per cada comandament de veu.

5.3 Muntatge

Pel muntatge final del projecte, s'ajuntaran totes les peces del projecte per formar un braç amb pinça.

Les parts que conformen el braç, al estar preparats per contenir un servomotor, no necessiten cap element d'unió amb aquest, ja que el motor queda encaixat a l'orifici dissenyat per aquest.

Al ser un prototip, el robot no està pensat per moure's a gran velocitat ni aguantar massa càrrega de pes. Llavors, no es considera necessari elements d'unió massa robustos. Per tant, per facilitar la construcció, es va decidir enganxar amb cola instantània la peça superior, i així poder muntar i desmuntar les parts del braç quan fos necessari.

Per la part de la pinça, al ser un disseny d'Internet, es detallava el procés de muntatge (veure Figura 5.2). L'únic que va ser necessari va ser les peces impreses en 3D amb el procés ja comentat, cargols de mètrica 3$times$16 i femelles de mètrica 3.

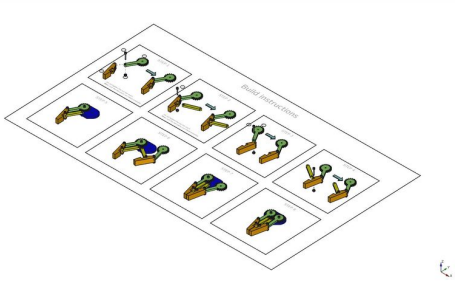

Figura 5.2: Instruccions de muntatge de la pinça

Per suportar el braç es va decidir utilitzar dues fustes, una més gran (de 167×120 mm) que tindria el braç, la placa Arduino i la placa de proves, i una altra més petita (de 120×40 mm) que tindria les entrades d'alimentació i USB, a més de suportar el mòdul de control de veu.

Per últim, a la Figura 5.3 es detalla l'esquema de connexions.

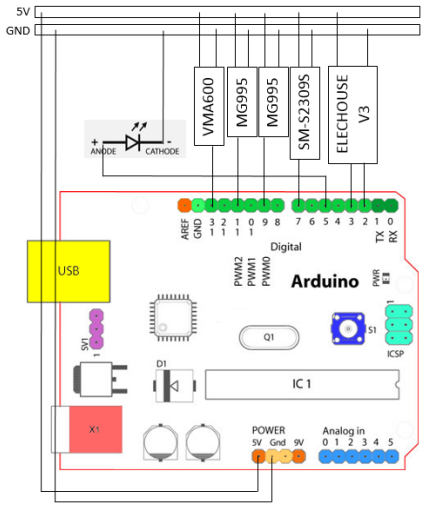

Figura 5.3: Esquema de connexions

Capítol 6

Experimentació

En aquest apartat, dissenyarem un seguit de situacions per veure com es comporta el robot. S'explicaran i es referenciaran als vídeos pujats a Internet perquè es puguin veure a qualsevol moment.

Com a sistema d'avís, abans i després de cada moviment s'encendrà un LED vermell per a que l'usuari sigui conscient que el braç està funcionant.

6.1 Aguantar un objecte

Aquest es un dels moviments o objectius que es tenia pensat per un *cobot*. Dins la pròpia experiència, moltes vegades es necessita una tercera mà a l'hora de fer algunes tasques, com per exemple soldar.

El moviment es basarà en anar amunt o avall utilitzant els comandaments *puja* o *a sota* respectivament, i juntament amb el comandament *tanca*, que fa tancar la pinça i així pot agafar l'objecte.

Es pot veure el moviment que segueix el braç robot a `https://youtu.be/Ogchbbc01es`

6.2 Moure un objecte

Un altre moviment que es pot comprovar és moure quelcom d'una posició a una altra.

Per fer-ho, es farà sostenir un objecte a qualsevol posició, sigui la inicial, amunt o avall, i utilitzant el comandament desitjat, es mourà el braç a aquella posició, tot mantenint l'objecte.

Per veure el moviment, l' enllaç `https://youtu.be/WDMxFrmwtow` mostra un exemple.

6.3 Roscar o desenroscar

Aquest moviment serveix per veure si el *cobot* és útil a l'hora d'ajudar a un operari a roscar o desenroscar una peça a la que hagi d'accedir un humà per la seva situació o risc associat.

Per efectuar els moviment, existeixen els comandaments *rosca* i *desenrosca*, especialment pensats per aquesta tasca. Aquests moviments es poden fer en qualsevol posició.

Per veure el moviment, s'haurà de seguir aquest enllaç: `https://youtu.be/B2Hi_zJVQ5Y`

6.4 Apagar

Mitjançant el comandament *apaga't*, el braç va a una posició recollida i desconnecta tots els servomotors, quedant apagat.

Es pot veure un exemple del moviment a l'enllaç `https://youtu.be/8GMgexOVcTg`

Capítol 7

Treball futur

En aquest capítol s'estudiarà el que passaria amb aquest projecte si es fiqués en un ambient laboral tal i com està ara, i com es podria millorar en el futur. En el cas que el prototip entrés tal com està a una fàbrica per utilitzar-lo com a *cobot*, es obvi que tindria diversos problemes.

El principal problema que es veu és el soroll. En un entorn d'una fàbrica el més normal és que hi hagi molt soroll extern. Com ja s'ha comprovat durant les proves al fer la programació, el mòdul utilitzat és bastant sensible al soroll extern.

Com a possible solució a aquest problema, es podria posar un filtre per evitar que el soroll molesti al reconeixement de veu. Això es podria solucionar amb un filtre passa baixos [6].

Aquests filtres funcionen amb un circuit RC (Figura 7.1) que es comporta de la següent manera: En arribar una freqüència baixa, el condensador donarà poca oposició al pas de la corrent, el que farà que la caiguda de tensió es doni a la resistència i, per tant, a la sortida.

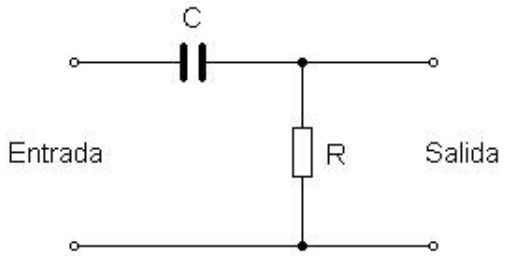

Figura 7.1: Circuit RC

Un altre problema seria la versatilitat del braç. Ara mateix és un braç de 2 graus de llibertat, però fàcilment podria ser de 3 graus de llibertat afegint un servomotor semblant al de les articulacions a la base. Així, es pot afegir el moviment a l'eix Z.

De la mateixa manera, l'existència d'aquest tercer eix faria que la inseguretat del braç augmentés. Es a dir, podria donar-se el cas que algú passés per davant del braç just quan es mogués, i això podria derivar en un accident. Per poder solucionar aquest problema, es podria afegir al braç sensors de proximitat, i mitjançant codi programar la distància de seguretat a la que es pararà el braç, semblant als sensors de proximitat que tenen els cotxes actuals i que ja estan incorporats als braços robots col·laboratius que estan al mercat.

Un altre problema que té el disseny existent és el cablejat. Tal com està dissenyat ara, no hi ha una bona gestió dels cables. Permet el moviment sense problemes, però en el moment que el moviment sigui més complex aquesta gestió del cablejat no funcionarà. Com a solució a aquest problema, es podria utilitzar un circuit fet amb plaques de prototipat de baquelita. En ser soldat, els cables no es mourien i s'evitaria el problema de que es surtin de la *protoboard*, com pot passar actualment. A més, es podria utilitzar una placa *Sensor Shield*, el que evitaria haver d'utilitzar la *protoboard*, ja que ella mateixa podria donar el voltatge necessari a tot el que es connecti. També es podria modificar el disseny del braç per tal que el cablejat pugui anar per dins del propi braç, i després tenir una caixa o semblant on estaria tot el cablejat. Això faria que els cables es moguessin juntament amb el braç i arreglaria aquest problema. La situació actual demostra la inexperiència en el disseny d'aquest tipus de braços.

Finalment, el braç funciona però una vegada fet, experimentant amb ell i veient altres dissenys, es pot veure que la part final del braç (el que seria la part colze-canell) és massa llarga, el que genera molt moment a tot el braç quan ha d'aguantar un objecte, o genera una tibantor dels cables innecessària. Llavors, per un disseny més òptim aquesta part hauria de ser més curta del que és actualment, com es pot veure en dissenys ja existents.

Capítol 8

Conclusions

En aquest apartat. es farà un petit resum de tot el que s'ha après durant la realització d'aquest treball, a més de fer un sumari de la finalització i el futur d'aquest projecte.

En aquest treball he tingut la oportunitat de treballar en elements de l'electrònica tan àmpliament utilitzats com *Arduino* i el seu entorn. He après també part de la programació C++, necessària per poder programar aquesta placa.

També, pel fet d'haver dissenyat el braç, he pogut experimentar el que es dissenyar quelcom. Com a reflexió, una vegada acabat el projecte m'he donat compte de les coses que he fet malament a l'hora de fer el disseny, i així penso que si hagués de fer un altre disseny ho faria d'una manera molt diferent a com s'ha realitzat aquest projecte.

Per la part objectiu del treball, el control per veu, he vist que és una eina útil pel control d'un *cobot*. Al poder donar-li ordres, ens estalviem utilitzar una consola per controlar-lo, i així es pot demanar tasques al robot mentre l'operari continua fent altres coses, o treballar conjuntament amb ell. Tot i així, aquesta idea està en una fase molt temprana del seu desenvolupament. Ara mateix només pot reaccionar a ordres molt senzilles, i per tant només pot fer tasques senzilles. A més, s'hauria d'aplicar, com ja s'ha comentat, filtres per tal de poder aïllar la veu d'un entorn amb molt soroll com és una fàbrica.

Tot i el que he comentat, crec que aquest treball ha sigut una bona manera de iniciar-se ala robòtica, l'electrònica i estudiar quelcom que pot ser molt útil en el futur.

Capítol 9

Pressupost

El projecte s'ha realitzat al llarg de 17 setmanes, des de la primera setmana de setembre de 2019 a la setmana del 8 de Febrer de 2020.

De forma resumida, es pot parlar de 3 grans blocs de treball :

- Estudi teòric dels conceptes relacionats amb el projecte. Això engloba estudiar les possibilitats que es tenien per fer el projecte, aprendre a utilitzar el llenguatge de l'elecció i, a més, el fons teòric del concepte del treball.

- Disseny del robot tant de forma física la seva programació.

- Construcció del robot i programació, afegint les proves del programa.

El temps aproximat que s'ha dedicat per setmana al projecte ha sigut d'unes 20 hores, fent a la vegada la part pràctica del projecte i la memòria.

En aquestes hores, a més, s'ha de tenir en compte que no totes costen de la mateixa manera. S'haurà de tenir en compte que les hores de recerca teòrica tindran el màxim cost, d'uns 30 € l'hora; les hores del projecte com a tal tindran un cost de 20 €; finalment, les hores necessàries per fer la memòria seran les mes barates, a 10 € l'hora. Per englobar el cost, ja que moltes vegades les feines s'han fet de forma simultània, tindrem en compte un preu mig de 20 €.

D'altra banda, s'ha de tenir en compte el cost energètic i de materials. Pel cost energètic, s'agafarà un cost mig de 0.0058 € hora. En el cas de l'Internet no tindré en compte el seu cost ja que és un preu fixe i no va per hores de consum.

Per tant, es desglossarà el cost de materials (Taula 9.1):

Material	Preu (€)
Placa *Arduino UNO*	7.40
ELECHOUSE V3	28.99
Protoboard	4
MG995 x2	14.88
VMA600	4.13
SM-S2309S	9.92
Font d'alimentació	10.39
Bobina de PLA	15-20
Fusta	3.50
Cargols i femelles	1.50
Sergent	4.61
Cost total del materials	**109.32**

Taula 9.1: Desglòs del cost dels materials. *No es té en compte el cost dels cables*

El desglòs de les hores (Taula 9.2) serà així:

Concepte	Dedicació	Cost horari (€/h)	Cost total (€)
Hores dedicades	340 h	20	6800
Electricitat	340 h	0.0058	1972
Cost total			**8772**

Taula 9.2: Taula de costos. *S'ha considerat una despesa energètica horària de 0.40 kWh i un preu de 0.115 € el kWh.*

Finalment, doncs, el pressupost final del projecte serà de **8881.32 €**

Capítol 10

Impacte Ambiental

En relació a l'impacte ambiental, s'ha de tenir en compte l'impacte que té la despesa elèctrica de l'ordinador amb el que s'ha treballat i la màquina d'impressió 3D. Observant la taula 9.2 es pot veure que s'ha consumit aproximadament 136 kWh d'energia provinents del propi ordinador, i en l'arxiu d'informació de la impressora 3D [11] es pot veure que ha consumit aproximadament 8.4 kWh. Tenint en compte un valor de mix elèctric peninsular de 321 g CO_2/kWh, la petjada de carboni associada al consum elèctric resulta de 46.35 kg CO_2, aproximadament. A més, s'haurà de tenir en compte l'ús del plàstic per imprimir. Al ser un prototip, s'ha utilitzat poc material per construir-lo. A més, el plàstic utilitzar ha sigut PLA, que deriva de matèries primes i renovables, no de recursos fòssils [1]. Per tant, tot i no ser completament ecològic, és una de les millors opcions.

Finalment, s'ha de considerar que aquest és l'impacte ambiental d'un prototip. Si parléssim d'un robot col·laboratiu real, el resultat de l'estudi ambiental seria molt diferent.

Capítol 11

Agraïments

Voldria agrair a la meva amiga Eva Deltor, que m'ha ajudat amb els seus coneixements de robòtica i electrònica, especialment amb el mòdul de control de veu, aconsellar-me i encoratjar-me; al meu amic Gerard Valls, per ajudar-me en temes d'electrònica i fabricació additiva de les peces, a més d'aconsellar-me en com millorar; al meu pare, Jordi Fontal, per ajudar-me a la construcció de la base del braç; i al meu tutor Cecilio Angulo, que ha estat sempre que he necessitat ajuda amb el projecte.

Bibliografia

[1] 3Dnatives. www.3dnatives.com/es/ecologico-realmente-filamento-pla-230720192/.

[2] Aprendiendo Arduino. aprendiendoarduino.wordpress.com/2015/03/26/lenguaje-de-programacion-c/, 2015.

[3] Arduino. store.arduino.cc/arduino-uno-rev3.

[4] Creality. https://tinyurl.com/raxwdry.

[5] Createc. createc3d.com/altura-de-capa/, 2014.

[6] Ecured. www.ecured.cu/Filtro_pasa_bajos.

[7] Elechouse. www.elechouse.com/elechouse/index.php?main_page=product_info&products_id=2254.

[8] Electronica Eurogold. electronicaeurogold.com/cargadores-de-salida-fija/927-alimentador-conmutado-9v-2a-5-5x2-1mm.html.

[9] Electrónicos Caldas. www.electronicoscaldas.com/es/pic-dspic/16-pickit-3-programador-quemador-pic-usb.html.

[10] filament2print. Tècniques d'impressió,filament2print.com/es/blog/23_skirt-brim-raft.html, 2017.

[11] Gerard Valls. drive.google.com/open?id=1D8Bsy5pPCFzEbDqUQUR1VpRmArQE2Isf.

[12] Javier Isabel Hernández, Juan González. http://www.iearobotics.com/wiki/index.php?title=Freecad:_Pinza_mecanica.

[13] Leantec. leantec.es/tienda/micro-servo-sm-s2309s/.

[14] Panamahitek. panamahitek.com/que-es-y-como-funciona-un-servomotor/.

[15] Velleman. www.velleman.eu/products/view/?id=438232.

[16] Wikipedia. es.wikipedia.org/wiki/Arduino.

[17] Wikipedia. es.wikipedia.org/wiki/C%2B%2B.

[18] Wikipedia. es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n).

[19] Wikipedia. es.wikipedia.org/wiki/Placa_de_pruebas.

[20] Wikipedia. es.wikipedia.org/wiki/Python.

[21] Wikipedia. es.wikipedia.org/wiki/Raspberry_Pi.

[22] Wikipedia. es.wikipedia.org/wiki/Rob%C3%B3tica.

[23] Wikipedia. s.wikipedia.org/wiki/Cobot.

[24] Wikipedia. www.electronicoscaldas.com/es/motores-y-servos/608-servo-motor-mg995.html.

Apèndix A

Codi *Arduino*

```
#include <SoftwareSerial.h>
#include "VoiceRecognitionV3.h"
#include <Servo.h>


VR myVR(2,3);
Servo base;
Servo base_codo;
Servo codo_muneca;
Servo pinza;
uint8_t records[7]; // save record
uint8_t buf[64];

int pos_base = 83;
int pos_base_codo = 53;
int pos_codo_muneca = 90;
int vuelta = 0;
const int listenLED = 5;


#define a_sota     (0)
#define puja    (1)
#define tanca    (2)
#define obre    (3)
#define rosca    (4)
#define apagat    (5)
```

```
#define desenrosca    (6)

void printSignature(uint8_t *buf, int len)
{
  int i;
  for(i=0; i<len; i++){
    if(buf[i]>0x19 && buf[i]<0x7F){
      Serial.write(buf[i]);
    }
    else{
      Serial.print("[");
      Serial.print(buf[i], HEX);
      Serial.print("]");
    }
  }
}

/**
  @brief    Print signature, if the character is invisible,
            print hexible value instead.
  @param    buf  --> VR module return value when voice is
     recognized.
              buf[0]  --> Group mode(FF: None Group, 0x8n: User,
                  0x0n:System
              buf[1]  --> number of record which is recognized.
              buf[2]  --> Recognizer index(position) value of
                 the recognized record.
              buf[3]  --> Signature length
              buf[4]~buf[n] --> Signature
*/
void printVR(uint8_t *buf)
{
  Serial.println("VR Index\tGroup\tRecordNum\tSignature");

  Serial.print(buf[2], DEC);
  Serial.print("\t\t");

  if(buf[0] == 0xFF){
    Serial.print("NONE");
```

```
  }
  else if(buf[0]&0x80){
    Serial.print("UG ");
    Serial.print(buf[0]&(~0x80), DEC);
  }
  else{
    Serial.print("SG ");
    Serial.print(buf[0], DEC);
  }
  Serial.print("\t");

  Serial.print(buf[1], DEC);
  Serial.print("\t\t");
  if(buf[3]>0){
    printSignature(buf+4, buf[3]);
  }
  else{
    Serial.print("NONE");
  }
  Serial.println("\r\n");
}

void setup()
{
  /** initialize */
  myVR.begin(9600);
  base.attach(9);
  base_codo.attach(11);
  codo_muneca.attach(13);
  pinza.attach(7);
  base.write(83);
  base_codo.write(53);
  codo_muneca.write(180);
  pinza.write(95);
  Serial.begin(115200);
  Serial.println("Elechouse Voice Recognition V3 Module\r\
      nControl LED sample");
  pinMode(listenLED, HIGH);
```

```
  if(myVR.clear() == 0){
    Serial.println("Recognizer cleared.");
  }else{
    Serial.println("Not find VoiceRecognitionModule.");
    Serial.println("Please check connection and restart Arduino.
        ");
    while(1);
  }

  if(myVR.load((uint8_t)a_sota) >= 0){
    Serial.println("a_sota loaded");
  }

   if(myVR.load((uint8_t)puja) >= 0){
    Serial.println("puja loaded");
  }

   if(myVR.load((uint8_t)tanca) >= 0){
    Serial.println("tanca loaded");
  }

   if(myVR.load((uint8_t)obre) >= 0){
    Serial.println("obre loaded");
  }

   if(myVR.load((uint8_t)rosca) >= 0){
    Serial.println("rosca loaded");
  }

     if(myVR.load((uint8_t)apagat) >= 0){
    Serial.println("apagat loaded");
  }
       if(myVR.load((uint8_t)desenrosca) >= 0){
    Serial.println("desenrosca loaded");
  }
}

void loop()
```

```
{
  int ret;
  ret = myVR.recognize(buf, 50);
  if(ret>0){
    switch(buf[1]){

      case a_sota:
        digitalWrite(listenLED, HIGH);
        delay(500);
        digitalWrite(listenLED, LOW);
        if (pos_base_codo <=110) {
        while(pos_base_codo<=110){
        base_codo.write(pos_base_codo);
        delay(10);
        pos_base_codo++;
        }
        }
        else{
        while(pos_base_codo>=110){
        base_codo.write(pos_base_codo);
        delay(10);
        pos_base_codo--;
        }
        }
        delay(750);
        if (pos_base >=60) {
        while(pos_base>=60){
        base.write(pos_base);
        delay(10);
        pos_base--;
        }
        }
        else {
        while(pos_base<=60){
        base.write(pos_base);
        delay(10);
        pos_base++;
        }
        }
```

```
    pos_base = 60;
    pos_base_codo = 110;
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    break;

  case puja:
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    if (pos_base <= 130) {
    while(pos_base<=130){
    base.write(pos_base);
    delay(10);
    pos_base++;
    }
    }
    else {
    while(pos_base>=130){
    base.write(pos_base);
    delay(10);
    pos_base--;
    }
    }
    delay(750);
    if (pos_base_codo <=170) {
    while(pos_base_codo<=170){
    base_codo.write(pos_base_codo);
    delay(10);
    pos_base_codo++;
    }
    }
    else {
    while(pos_base_codo>=170){
    base_codo.write(pos_base_codo);
    delay(10);
    pos_base_codo--;
    }
```

```
    }
    pos_base = 130;
    pos_base_codo = 170;
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    break;

  case tanca:
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    pinza.write(95);
    delay(500);
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    break;

  case obre:
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    pinza.write(145);
    delay(500);
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    break;

  case desenrosca:
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    while (vuelta != 4){
    delay(500);
    pinza.write(95);
    delay(500);
    codo_muneca.write(0);
```

```
    delay(500);
    pinza.write(145);
    delay(500);
    codo_muneca.write(180);
    vuelta++;
    }
    vuelta = 0;
    codo_muneca.write(180);
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    break;

  case rosca:
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    while (vuelta != 4){
    delay(500);
    codo_muneca.write(0);
    delay(500);
    pinza.write(95);
    delay(500);
    codo_muneca.write(180);
    delay(500);
    pinza.write(145);
    vuelta++;
    }
    vuelta = 0;
    codo_muneca.write(180);
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
    break;

  case apagat:
    digitalWrite(listenLED, HIGH);
    delay(500);
    digitalWrite(listenLED, LOW);
```

```
codo_muneca.write(180);
pinza.write(95);
codo_muneca.write(180);
if (pos_base_codo <=0) {
while(pos_base_codo<=0){
base_codo.write(pos_base_codo);
delay(10);
pos_base_codo++;
}
}
else{
}
while(pos_base_codo>=0){
base_codo.write(pos_base_codo);
delay(10);
pos_base_codo--;
}
delay(750);
if (pos_base >=20) {
while(pos_base>=20){
base.write(pos_base);
delay(10);
pos_base--;
}
}
else {
while(pos_base<=20){
base.write(pos_base);
delay(10);
pos_base++;
}
}
base.detach();
base_codo.detach();
codo_muneca.detach();
pinza.detach();
digitalWrite(listenLED, HIGH);
delay(500);
digitalWrite(listenLED, LOW);
```

```
            break;

        }
        /** voice recognized */
        printVR(buf);
    }
}
```